Describing Customizable Products on the Web of Data

François-Paul Servant
Renault
13 avenue Paul Langevin
92359 Plessis Robinson, France
francois-paul.servant@renault.com

Edouard Chevalier
Renault
13 avenue Paul Langevin
92359 Plessis Robinson, France
edouard.chevalier@renault.com

ABSTRACT
Exposing data about customizable products is a challenging issue, because of the number of features and options customers can choose from, and of the intricate constraints between them. However, the configuration process, by which the customer makes her choice, one step at a time, is a graph traversal among partially defined products; that is Linked Data browsing. This natural yet fruitful abstraction for product customization hides complexity from the client agent, and allows corporations to publish descriptions of their ranges of products, in their own terms. To open these ranges to comparison, corporate vocabularies have to be linked to known entities in the LOD cloud; the creation of sharable thesauri is discussed.

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous

Keywords
Linked Data, Structured eCommerce data, Configuration, GoodRelations, Automotive

1. INTRODUCTION
This work aims at defining a framework for the publishing and for the consumption of data describing ranges of customizable products.

In order to improve e-business performance and visibility, an increasing number of manufacturers and vendors publish product data on the web, relying on vocabularies such as GoodRelations[1] and the schema.org initiative. This makes their websites usable by client applications and improves the accuracy of search engines, with interesting results for products such as books or music: search results list actual products rather than mere web pages, and include links to commercial offers with price, ratings, etc.

Extending the publishing of data to customizable products, such as cars, raises challenging issues, which we study in this paper.

In industries practicing "Build to Order" of customizable products, ranges are huge, because of the number of features and options a customer can choose from: more than 10^{20} different cars are for sale at Renault. Many constraints between specifications reduce the number of valid combinations, making the range difficult to describe (we use the term "specification" to refer to the various characteristics of a product a customer can choose from); if every combination of distinct specifications were possible, there would be 10^{25} different Renault cars, not our mere 10^{20}. In other words, by picking one choice at random, independently for every specification, you would have only one chance out of 100,000 to define a car Renault actually sells. Automatic reasoning is required to handle the constraints and is a computationally hard task. Such range definitions can be published on the Web, including the constraints, by means of Semantic Web languages [4]. But it would not bring many practical results in the near future, as one cannot expect strong reasoning capabilities from client agents.

Difficult to specify and hard to manipulate as they are, these ranges of products are nevertheless described rather effectively, for human users, by means of dedicated web applications called configurators. A configurator helps a user to interactively define a product step by step, one choice after the other, each step describing a valid "Partially Defined Product" (PDP), with a start price and a list of remaining choices given all previous selections.

In a previous paper, "Product Customization as Linked Data" [3], we have shown that product configuration can be seen as a linked data application. The main idea is the modeling of the configuration process as the traversal of a graph whose nodes are PDPs, or "Configurations", each configuration linking to those that refine it, until completion. By identifying each PDP with a URI returning, among other-relevant information, the list of the PDPs it is linked to, we turn the description of the range to Linked Data. Identifying the PDPs with URIs opens fruitful perspectives to e-business, such as the easy sharing of configurations between online and in-house applications, devices and media.

A simple and generic ontology[1] describes the classes and properties involved in this modeling of the configuration pro-

[1]http://purl.org/configurationontology

cess as Linked Data. This ontology is generic: it is domain independent.

Renault uses this modeling to publish the data about its range[2]: hundred billions of billions (10^{20}) different cars, fully described in RDF.

These descriptions, however, do not contain any link to the rest of the world, that is, to existing vocabularies of the linked data cloud that could provide shared understanding of the terms. The only available information from which some common understanding could be built are the labels included with the descriptions. But what dataset could provide URIs for the many terms needed? You may find, say, "Automatic gearbox" or "Diesel" in DBpedia; what about "Panoramic electric sunroof", "dual zone climate control", or "105g CO2/km", though?

Therefore, more work needs to be done to open ranges of similar products, from different constructors, to comparison; work by the producers of data, by the clients, or by third parties, to link the corporate vocabularies used in the description of those ranges to entities defined in sharable thesauri.

The ultimate goal is to allow customers, not only to search for partially defined products on the web - "find a small diesel car with an automatic gearbox, a sunroof and MP3 connection plug for less than 15,000 euros" - but also to choose themselves on which scale to compare results, e. g. "available navigation systems plus CO2 emission levels".

This document is structured as follows: section 2 lists related works. In section 3, we come back to the modeling of the configuration process as linked data. This was discussed in our earlier paper, but as it is the cornerstone of our work, it cannot be avoided here. Section 4 explains how this integrates with GoodRelations. Section 5 deals with the shared understanding of terms, and the design of vocabularies dedicated to the definition of specifications.

2. RELATED WORK

The GoodRelations (GR) ontology, "the web vocabulary for e-commerce", has become a de-facto standard for the publishing of e-business data on the web, a status that has been recently reinforced by its integration to the Schema.org collection of schemas that is promoted by the major search engines.

GR allows the description of product offerings. It has been designed to be used in combination with additional ontologies for product types and their properties. This makes perfect sense (separation of concerns). Several such GR compliant vocabularies have been developed, including by Hepp Research, the creator of GoodRelations.

A study about the usage of GR, in 2011, notes however that "the lack of available product ontologies and product master datasheets is impeding the creation of a semantically interlinked eCommerce Web" [2]. This conclusion should probably be reexamined as new GR compliant vocabularies have been published. The Product Types Ontology (PTO) for instance is a service for providing GR compatible OWL DL class definitions for the 300,000 types of real-world products or services that have an entry in the English version of Wikipedia. Schema.org has been updated with the "additionalType" property, precisely to allow the use of PTO classes to describe products.

Let us now turn to the situation of customizable products. GR doesn't seem to address the case of "partially defined products". At least, it does not make any explicit mention of them. It could be argued that GR's ProductOrServiceModel class, an abstraction used to model "prototypes" of products, along with the ability to derive descriptions of actual products from it, could be used for that purpose, but it requires non standard reasoning. The Vehicle Sales Ontology (VSO), a vocabulary by Hepp Research, seems to define properties only meant to be used in describing completely defined products, in spite of their focus on a domain where product customization is an important topic.

In the exact context of our work - that is, the grey area where product configuration meets e-Commerce applications of Linked Data - the main contribution that we know of is Volkswagen's "Car Option Ontology"[3] (COO), an extension to VSO. Their approach is different from the Configuration Ontology we advocate: they include the constraints between options in the publication, in a proprietary vocabulary. This puts the burden of the reasoning on the client; however, simple client agents do not have reasoning capabilities.

3. CONFIGURATION AS LINKED DATA

This was the subject of our previous paper, which we shall only summarize here. For a complete treatment, please refer to the original work[3].

3.1 Principles

As ranges of customizable products are too huge to be enumerated, they are defined in intention. The description of a family of similar products (typically those of the same "model") is based on a "lexicon", i.e., a set of variables representing the relevant descriptive attributes: body type, type of fuel, color, etc. In a completely defined product, any of these variables is assigned one value and one only. Then a set of constraints restricts the possible combinations of specifications. The definition of a range of customizable products is therefore a Constraint Satisfaction Problem (CSP) - something which is well known to be computationally hard.

A configurator application is the main way of presenting such complex range of customizable products to customers. It is a decision support tool that guides users to desirable - and valid - product configurations.

Typically, the user is presented with successive choices in a way that she cannot choose incompatible specifications. Each successive state of the configuration process is characterized by the specifications selected so far. A configuration engine is responsible for ensuring that only valid choices

[2]http://{de,es,it,fr,uk}.co.rplug.renault.com/docs
Quick start guide (live tutorial and js configurator) at http://purl.org/configurationontology/quickstart

[3]http://purl.org/coo/ns

are presented at each step. In most of the configurator applications each step describes therefore a valid partially defined product, in the sense that it can be completed, without changing any of the current selections, into an existing fully specified product, which can be ordered. We call "Configuration" any such valid, partially defined product: in other words, any state of the configuration process.

Such an application can therefore be implemented as a GUI over a REST service which takes the description of a valid configuration (the list of the specifications already selected) as input parameter, something like :

```
confService?chosenSpec=spec1&chosenSpec=spec2&...
                                                (1)
```

and returns the next list of specifications to be chosen from, all guaranteed to be compatible with the input. Choosing one of them is then just a matter of adding it to the list of the "chosenSpec" query parameters and of getting the updated state of the configuration process.

Note that a query such as (1) identifies a configuration, and can be used as a URI for the configuration in question; or, more precisely, redirect to an actual URI of it; therefore, we can improve the service by making it return the URI of the linked configuration along with each compatible specification: the representation of the configuration resource then contains a list of couples (compatible Specification, linked Configuration).

Such a service makes it easy to implement a configurator application: accessing a configuration URI returns the data needed to build the corresponding web page: basically a list of links to the next configurations.

Most if not all configurator application on the web could be (re-)implemented this way: it is just a matter of wrapping the configuration engine in a REST service that provides the data needed to generate the HTML.

The template of the service (1) can also be used as a simple querying API. Mind however that any combination of specifications may not be valid. The service should detect such invalid conjunctions and return a 404 Not found HTTP error. Only configuration engines that support free order can provide such functionality in every circumstance.

3.2 "Configuration as Linked Data" ontology

The "Configuration as Linked Data" ontology (COLD)[4] describes the classes and properties involved in the modeling of the configuration process as Linked Data. It is really simple, with three main classes (Configuration, Specification and ConfigurationLink), and a few properties that models the state of a specification with respect to a given configuration: is it chosen? implied? possible? etc. Here is an example Configuration:

```
foo:aConf cold:chosenSpec foo:Model1, foo:Diesel ;
cold:impliedSpec foo:ClimateControl;
cold:possible [a ConfigurationLink ;
cold:specToBeAdded foo:Sunroof;
cold:linkedConf foo:aConfWithSunroof].
```

[4]http://purl.org/configurationontology

3.3 Summary

Salient points in the above exposition are:

- configurations are first class objects, identified by URIs, and they can therefore easily be shared between applications.

- The complexity of the range is hidden from the client. No reasoning capability is required from the client agent.

- Ranges can be crawled, either starting from the root of the dataset or from any configuration, and following links whose semantics is precisely defined in the ConfigurationLink class. Search engines are provided with enough information to customize their strategies: they can choose which links they follow (not all specifications are of equal interest: the sunroof, the MP3 connector, etc. are probably more important - for a customer as well as for a search engine - than, say, the color of the ashtray.)

- The approach is domain independent.

4. INTEGRATION TO GOODRELATIONS

4.1 Product and commercial offer

A configuration mainly describes a Partially Defined Product. As such, in GoodRelations terms, a Configuration is a gr:ProductOrServiceModel: "an intangible entity that specifies some characteristics of a group of similar, usually mass-produced products, in the sense of a prototype." The suffix "Model" may seem misleading when used for a Configuration, as it suggests something such as "Ford T", and not "Ford T with climate control and MP3 connection plug" (itself not a completely defined product - you still can choose, well, the color).

On the other hand, a configuration has a price. It can be seen as a commercial offer, or the expression of a customer's wish list. It could therefore be considered as a gr:Offering as well. These two GR classes being disjoint, however, Configuration cannot be subClass of both.

4.2 gr:ProductFeature

GoodRelations has announced a new pattern for the description of products "that allows publishing arbitrary property-value pairs for product features"[5].

As of this writing, this is not available, and the online documentation is limited to a few examples, but it looks like the gr:ProductFeature class is semantically very close to our Specifications and could be used to represent them (a "Specification" is a value of a "ConfigurationVariable": it is a property-value pair). We therefore strongly support the idea, we hope that it will allow to fully support specifications, and we look forward to seeing it actually integrated to GR.

Of special interest are the motivations given for the introduction of this new pattern: "for publishing product features from shop sites and manufacturer data sheets, it is often too much of a burden to map the available product data into a

[5]http://wiki.goodrelations-vocabulary.org/Documentation /Product_features

standardized products or services ontology and its types and properties... With that feature, a shop owner could easily mark-up product feature tables and preserve as much data and as much data structure and other meta-data as possible without the need to cleanse and lift the data before publishing it."

Indeed, it is very important to allow for the publishing of data "as they are", without requiring preliminary work, and to decouple this initial publishing and the enhancement of the data. This is exactly what our approach allows, as will be argued below.

5. BUILDING SHARED UNDERSTANDING

The "Configuration as Linked Data" ontology is generic: it does not depend on the variables and specifications that define a product, and it allows a publisher to use its own terms in the descriptions. This is a very important point, because:

- the whole purpose of the configuration process is to come out with an order for a completely defined product, which implies its definition in the manufacturing company's terms,

- no precision is lost, in contrast to what would happen if we had to map to a different vocabulary

- it makes it possible to publish the data as they are in the publisher's systems, with no additional cost.

If only for these three reasons, the web of product data will be based on the publishing of data that use corporate vocabularies (in particular, no vendor will accept to downgrade the description of its product for the sole purpose of making it comparable to others' products). This leaves unresolved the question of the shared understanding which is necessary to, for instance, compare ranges of similar products; shared understanding can only be achieved through a deliberate and methodical effort to enhance those data. Various techniques can be used. For instance, labels can be used to (try to) automatically recognize known entities, or they can be used in SPARQL queries. In the end however, at least in the Linked Data context in which we are interested, shared understanding is achieved by linking to the Linked Data cloud, that is, by rooting the definition of entities through links to existing terms in well established vocabularies.

The question becomes, then: do URIs already exist for them? or can we automatically derive their URIs from preexisting ones, the way the "Product Types Ontology" derive URIs for products from DBpedia URIs? No doubt DBpedia will provide such terms as dbpedia:Diesel, or dbpedia:Sunroof, for instance; however most of what we need will be missing. Wikipedia pages may describe types of products and specifications, but it is clearly not the purpose of an encyclopedia to provide URIs for this kind of things, and it would be abusive to add pages to wikipedia for the sole purpose of creating these URIs.

The Configuration Ontology provides a generic framework that must be complemented with dedicated, domain dependent vocabularies providing the term definitions. These vocabularies, ideally maintained by a community, would establish the links to terms from DBpedia or other datasets in the LOD cloud, where such terms exist. They should also be as usable by organizations such as automotive manufacturers, that produce lots of data describing whole ranges of products, as by individuals selling their used cars (arguably, a single car is not a customizable product, but a vocabulary intended to describe a whole range should work, a fortiori, when restricted to the case of a single product in the range). The design of such a vocabulary should make it as easy as possible to enhance "raw data" i. e., data not cleaned up prior to publishing, in a process that do not require any impact on the source systems.

In the remaining of this text, we study some consequences of these requirements, and we express recommendations, based on practical concerns. These recommendations may seem obvious, but in the automotive field, the vocabularies we are acquainted with do not follow them, which prevented us from reusing them to increase the understandability of our published data.

Actually, our main point is simply to advocate a shift from vocabularies aimed at describing products, to vocabularies aimed at describing specifications.

5.1 Supporting Partially Defined Products

Whatever ontology we choose, be it COLD or not, a PDP is more than a Completely Defined Product with some properties left undocumented. This puts additional constraints on the design of the vocabulary.

Let's take an example to see the consequences. The "Vehicle Sales Ontology" (VSO) is a vocabulary created by Hepp Research to complement GoodRelations. It defines some properties, such "fuelType", to describe vehicles. The range of "fuelType" is the "FuelTypeValue" class, and VSO encourages the writing of statements such as:

```
foo:aCar vso:fuelType dbpedia:Diesel.
```

It is a very common pattern, however it does not work in our case, or at least not well. This is because the description of PDP requires the ability to state more than one kind of relation between a PDP and a given Specification: for a given partially defined car, the fuel type may have been chosen, or it may be implied, or we may want to say that both diesel and gazoline are possible values, but that electricity is not, etc. This is simply not possible with the vso:fuelType property.

Actually, the above statement conflates two pieces of information: the fact that dbpedia:Diesel is a FuelTypeValue, inferred from the definition of the range of vso:fuelType, and the fact that aCar is a diesel one. However, the former could have been stated once and for all:

```
dbpedia:Diesel a vso:FuelTypeValue.
```

to be able, then, to just write something like

```
foo:aCar :hasSpec dbpedia:Diesel.
```

to convey the same actual information in the result. Surely, no one will understand it as meaning Diesel is the color of that car! Seen in that light, statements such as "`foo:aCar vso:fuelType dbpedia:Diesel.`" are a bit pleonastic. In other words, we can forget the vso:fuelType property, and only retain the vso:FuelTypeValue class. This gets us back a degree of freedom in the triples to include the extra information we need to handle PDPs and to state, for instance, with the COLD ontology:

```
foo:aConf1 cold:chosenSpec dbpedia:Diesel.
foo:aConf2 cold:possible [a ConfigurationLink ;
cold:specToBeAdded dbpedia:Diesel;
cold:linkedConf foo:aDieselConf].
```

Also note that, even in the case of completely defined products, using properties such as vso:fuelType makes it costly to enhance raw data. Indeed, for every statement such as

```
foo:aCar :hasSpec dbpedia:Diesel.
```

we need to create a new statement using vso:fuelType; whereas it should have been enough, here again, to state dbpedia:Diesel is a FuelTypeValue to enhance all statements involving dbpedia:Diesel at once.

So, instead of defining multiple properties, it is better to define classes of Specifications. There is a small cost, however. When querying the description of a car for its fuel type, instead of just

```
SELECT ?x WHERE {foo:aCar vso:fuelType ?x.  }
```

we have to write:

```
SELECT ?x WHERE {
foo:aCar :hasSpec ?x.
?x a vso:FuelTypeValue.
}
```

5.2 Supporting hierarchies of terms

It is important that terms may be defined somewhat hierarchically, in order to allow some inferencing: a car with, say, an :ElectricSunroof, has a :Sunroof. This is necessary for search as well as for comparator applications. SKOS hierarchies could be used, but we tend to prefer a hierarchy of RDF-S or OWL classes: the semantics of subClassOf is well adapted to describe a hierarchy of classes of Specifications, tools are probably more widely available, and we expect little benefit from the less formal semantics of SKOS in that particular application. Note, however, that the vocabulary would define classes of Specifications, then, rather than instances, so that one should write statements such as: `foo:aCar :hasSpec [a :Diesel].`

5.3 Definition of values

For values, VSO recommends to use DBpedia URIs (or to mint new ones). We believe a vocabulary about vehicles should provide lists of terms - it should provide for instance :Diesel, and ensure its grounding in "the real world" by linking it to dbpedia:Diesel. To this purpose, it is better to avoid owl:sameAs statements, and we advocate the use of a dedicated property that would mean "links a class of Specification to something that can be interpreted as identifying the class in question" (the "Product Types Ontology" just uses rdfs:seeAlso for this kind of link).

Anyway, assuming a vocabulary asserting something like:

```
:FuelType rdfs:subClassOf cold:Specification.
:FuelType owl:equivalentClass vso:FuelTypeValue.
:Diesel rdfs:subClassOf :FuelType.
:Diesel rdfs:seeAlso dbpedia:Diesel.
```

then, to enhance a basic statement as a source of raw data would typically provide it:

```
foo:aCar :hasSpec foo:Diesel.
```

will be a simple matter of stating:

```
foo:Diesel a :Diesel.
```

Which can be done without any impact on existing systems, and improves all statements involving foo:Diesel at once. Is it not nice?

6. CONCLUSIONS

Data about customizable products can be published effectively as Linked Data. Most, if not all, configurator applications on the web could be modified with relative ease, to publish data that way. It gets us accurate descriptions of complex ranges of products, which can be crawled by simple agents: all reasoning takes place inside the service publishing the data, its complexity hidden from the clients. For search engines, the number of configurations is challenging - we added more than 10^{20} of them to the web of data - but the linked nature of the dataset should be sufficient to use it effectively. The unresolved issue is the public specification thesaurus, which we need to allow effective comparisons of ranges of similar products. We gave some recommendations for the creation of such vocabularies. We'd happily participate to a community initiative whose objective would be to create one such vocabulary for the automotive domain.

7. REFERENCES

[1] M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management.* Springer-Verlag, 2008.

[2] J. Ashraf, R. Cyganiak, S. O'Riain, and M. Hadzic. Open ebusiness ontology usage: Investigating community implementation of goodrelations. In *Proceedings of Linked Data On The Web Workshop*, 2011.

[3] E. Chevalier and F.-P. Servant. Product customization as linked data. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*, ESWC'12, pages 603–617, Berlin, Heidelberg, 2012. Springer-Verlag.

[4] F. Badra, F.-P. Servant and A Passant. A Semantic Web Representation of a Product Range Specification based on Constraint Satisfaction Problem in the Automotive Industry. OSEMA Workshop ESWC (2011) http://ceur-ws.org/Vol-748/paper4.pdf